Embedded System Design & Modeling

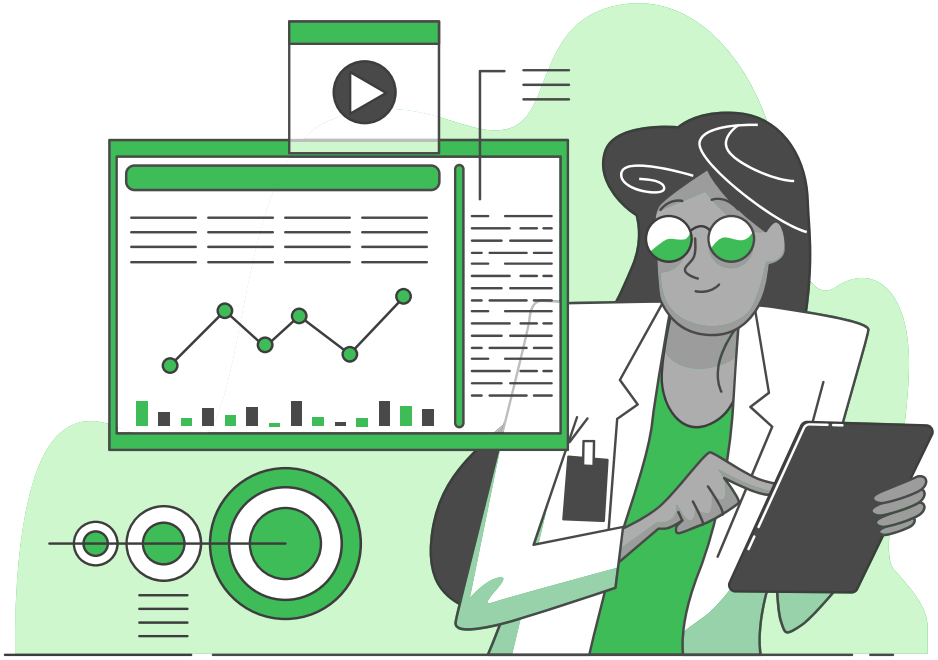
Environmental Monitoring System

By: Izabela Kacanja, Joni Brame, Melisa Tahiri, Klea Menga, Grei Meka

Sensitivity: Internal

Table of Contents

1. Objective
2. Requirement Analysis
3. Software development model
4. Use case modeling
5. System Design
6. Software Architecture Pattern Selection
7. Data Design & Behavioral Modeling
8. User Interface Mockups



Sensitivity: Internal

1. Objective

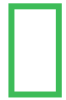
“The goal of this project is to develop a smart, real-time embedded system that monitors critical environmental parameters (temperature, humidity, and air quality), alerts users during hazardous conditions, and visualizes data through a user-friendly dashboard, while ensuring system reliability and efficiency.”

Sensitivity: Internal

Key Goals:

Measure **temperature**, **humidity**, and **air quality (CO_2, PM2.5)** in real-time.

Alert users when conditions exceed safe thresholds.



Visualize environmental data using an online dashboard.

Store data for historical analysis and future planning.

Use **low-cost, energy-efficient components** for continuous monitoring.

Sensitivity: Internal

2. Requirement Analysis:

1 Stakeholders

Environmental Scientists & Researchers: Interested in analyzing air quality, temperature, humidity, and noise levels for research and policy recommendations.

Pollution Control Authorities: Need real-time data to monitor and control pollution levels.

Smart City Planners: Integrate environmental data into smart city infrastructure.

General Public: Beneficiaries of improved environmental awareness and alerts.

Government Agencies: Use data for policy-making and enforcement.

System Developers: Responsible for hardware and software design.

Maintenance Staff: Ensure system functionality and sensor calibration.

Sensitivity: Internal

2 Assumptions and Constraints

The system will be **installed in a fixed location** for continuous monitoring.

Power supply (mains power or solar backup) will be available at the installation site.

Internet connectivity (Wi-Fi or LoRa) is available for cloud uploads.

Sensors are pre-calibrated for the specific environmental conditions.

Hardware components will have limited tolerance to extreme conditions (e.g., may not function in extreme weather or industrial environments).

The system will not support **mobile or wearable applications** in its current design.

Data storage capacity is limited by onboard

3 Functional Requirements

The system must:

- **Continuously monitor environmental parameters** including:
  - Temperature
  - Humidity
  - Air Quality (CO_2 levels, PM2.5)
  - Noise Levels
- **Display real-time data** on a local interface (e.g., LCD) and optionally via a web dashboard.
- **Generate alerts** when parameter values exceed predefined thresholds.
- **Store data locally** for historical analysis.
- **Provide periodic data uploads** to a cloud server for remote access (optional feature).
- **Operate in real-time** with an acceptable refresh rate (e.g., every 5 seconds).
- **Support calibration** for accurate sensor readings.

4 Non-Functional Requirements

The system should adhere to the following non-functional requirements:

- **Low Power Consumption**: Optimize hardware and firmware for energy efficiency.
- **Cost-Effectiveness**: Use affordable components without compromising reliability.
- **Accuracy**: Maintain data accuracy within ±5% tolerance.
- **Reliability**: System uptime should exceed 95% for continuous monitoring.
- **Scalability**: Allow the addition of new sensors (ozone, light intensity) in future iterations.
- **Security**: Ensure secure data transmission when connecting to cloud servers.
- **Environmental Robustness**: Operate reliably in typical environmental conditions (e.g., indoor/outdoor temperatures

Sensitivity:
Internal

Development Phases

- **Requirements Analysis**: Identifying environmental parameters (temperature, humidity, CO_2) and system functions.
- **System Design**: Design overall system structure (sensor selection, hardware block diagram, data flow).
- **Architectural Design**: Designing software architecture (layers: sensors, data processing, alerting, UI).
- **Module Design**: Designing individual modules (sensor drivers, data processing functions, LCD interface).
- **Code Implementation**: Writing and integrating embedded code (code for sensors and display).

Testing Phases

- **Unit Testing**: Testing individual modules like sensor data acquisition, LCD display, and alerts.
- **Integration Testing**: Testing the interaction between modules (sensors + data processing + LCD).
- **System Testing**: Testing the full system with all components (hardware + software) working together.
- **Acceptance Testing**: Verifying system meets stakeholder requirements (real-time monitoring, accuracy).

V-Model usage justification

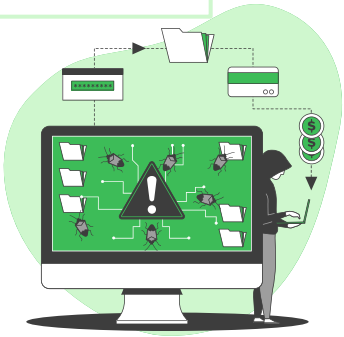
The V-Model is ideal because the system has clear, stable requirements for measuring environmental parameters. It combines embedded hardware and software, requiring early and continuous testing to ensure accurate integration.

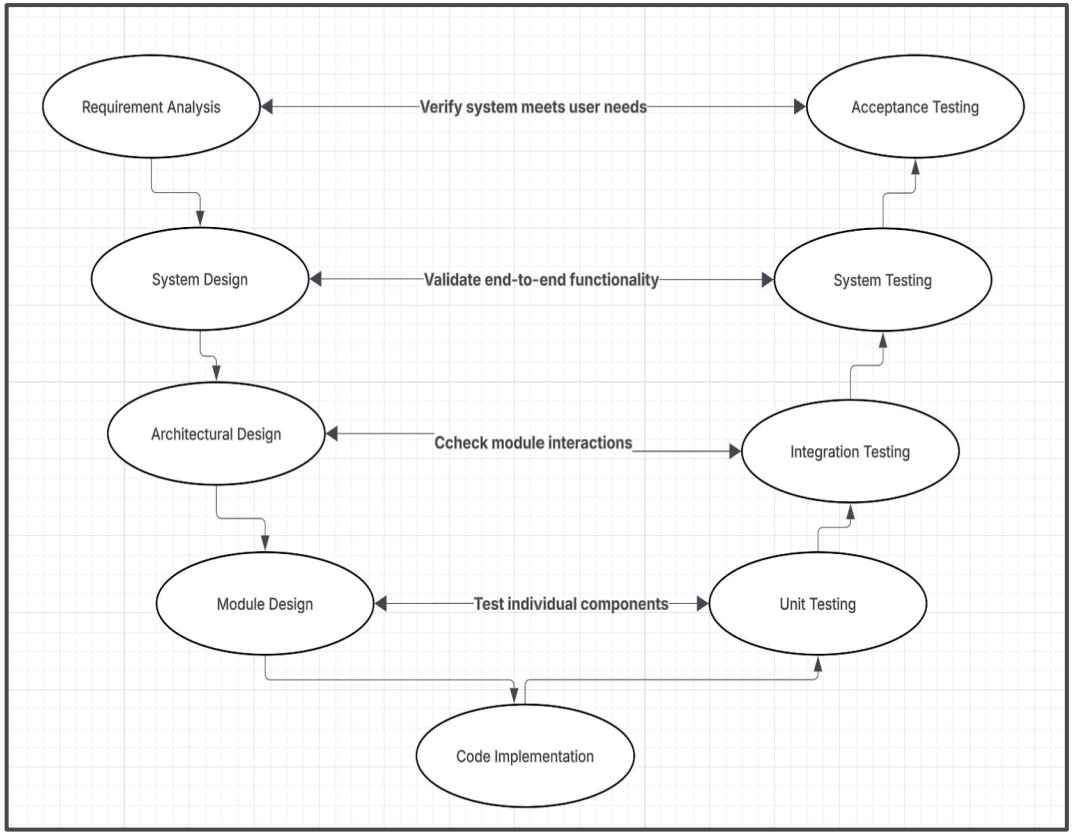
The model's structured approach helps catch errors early, improving quality and reliability. Given the project's moderate complexity and fixed goals, the V-Model provides a clear, efficient development and testing framework without needing iterative changes.

Sensitivity: Internal

3. Software development:

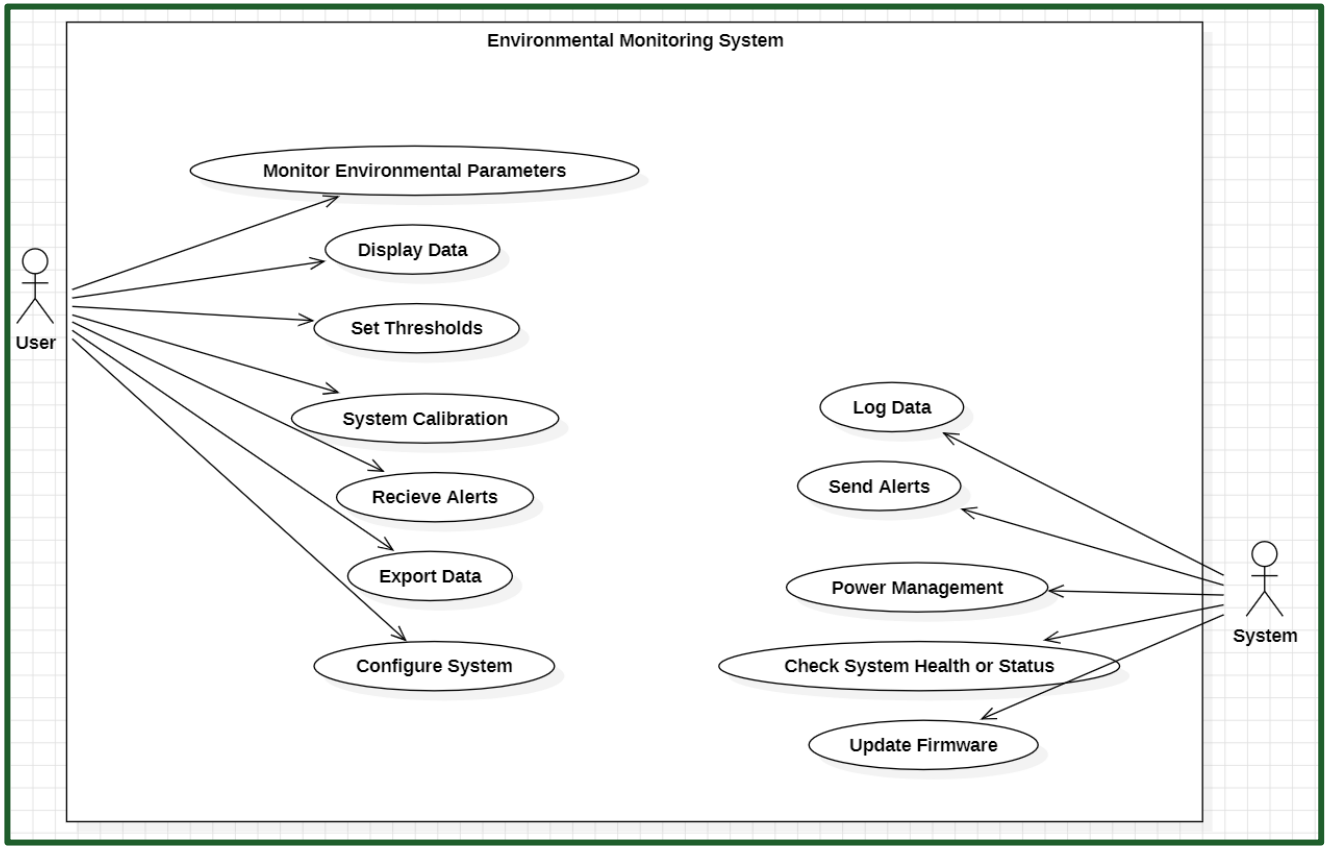
<u>V-Model</u> →Test driven approach with clear mapping between development and testing.





Sensitivity:
Internal

4. Use Case Diagram:



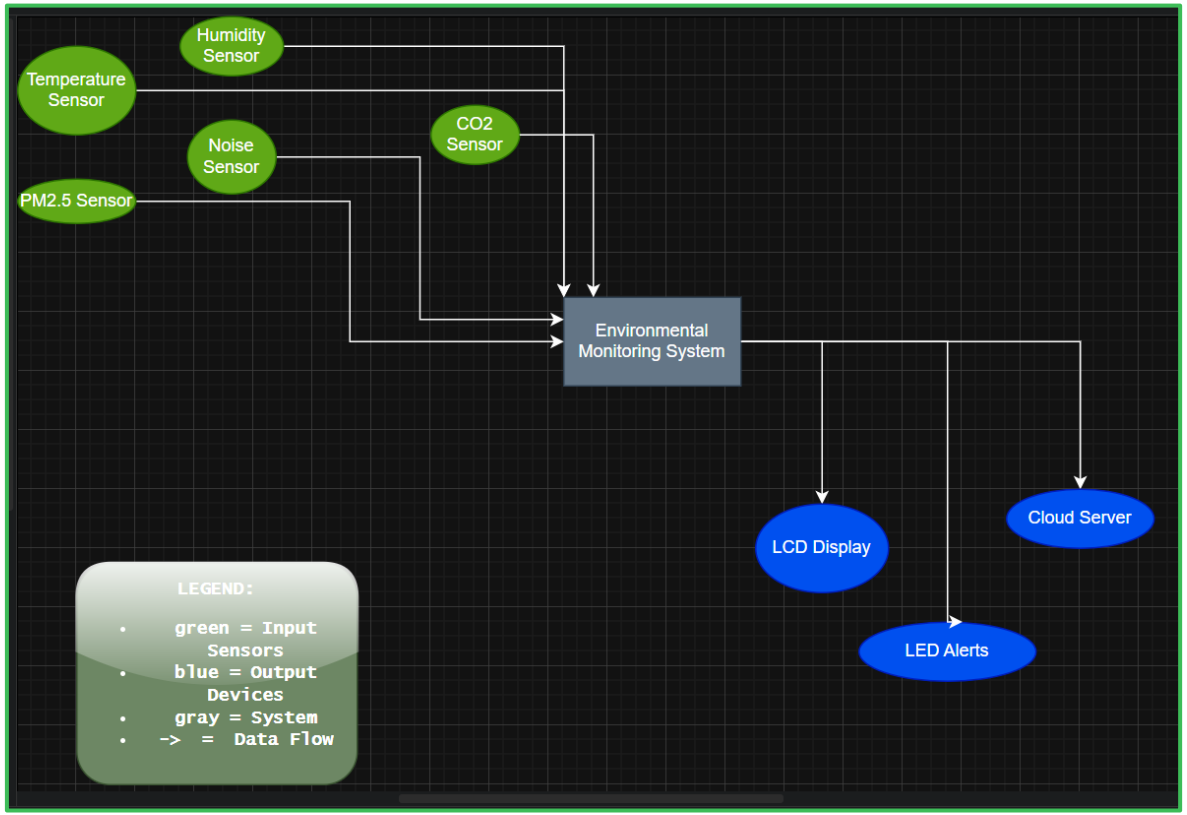
Sensitivity:
Internal

4. UC Tabular Specification:

System Requirement Name	Monitor Environmental Parameters
Description	Enables the user to observe real-time environmental data such as temperature, humidity and air quality.
Actors	User and System
Inputs	Sensor data from the environment.
Outputs	Displayed environmental parameters on UI
Steps to be taken	• User logs into the system • User selects the monitoring option • System retrieves data from the sensors • System processes and displays real-time data
Alternative Steps	• If sensor data is unavailable, show “No Data Available” warning • System retries data fetch after delay

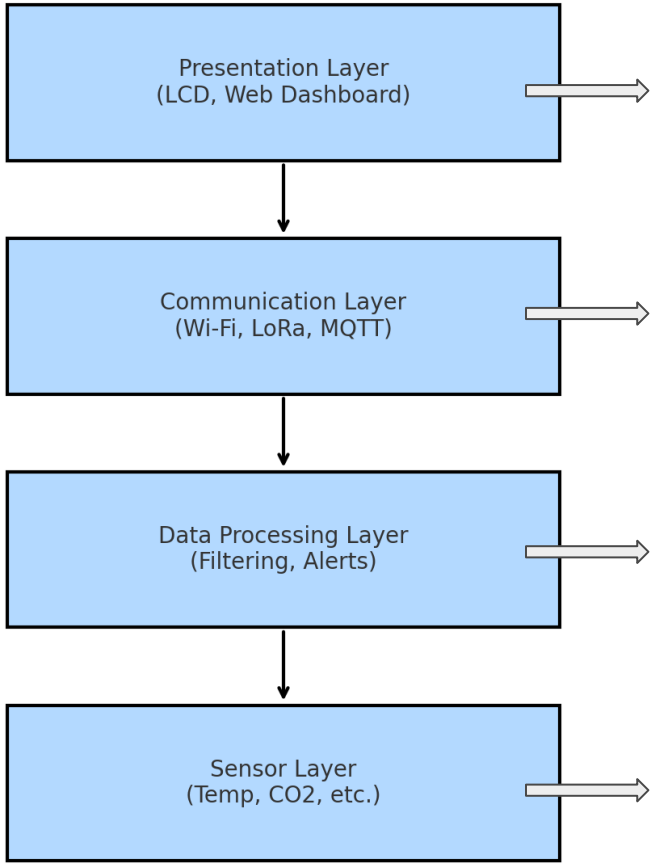
Sensitivity: Internal

5. System Design:



Sensitivity:
Internal

6. Software Architecture Pattern Selection:



Responsibilities:
-Display current readings and alerts
-Provide historical data visualization
Technologies:
-LCD/OLED Display (local output)
-Web Dashboard (HTML, CSS, JavaScript, Chart.js or ThingSpeak)

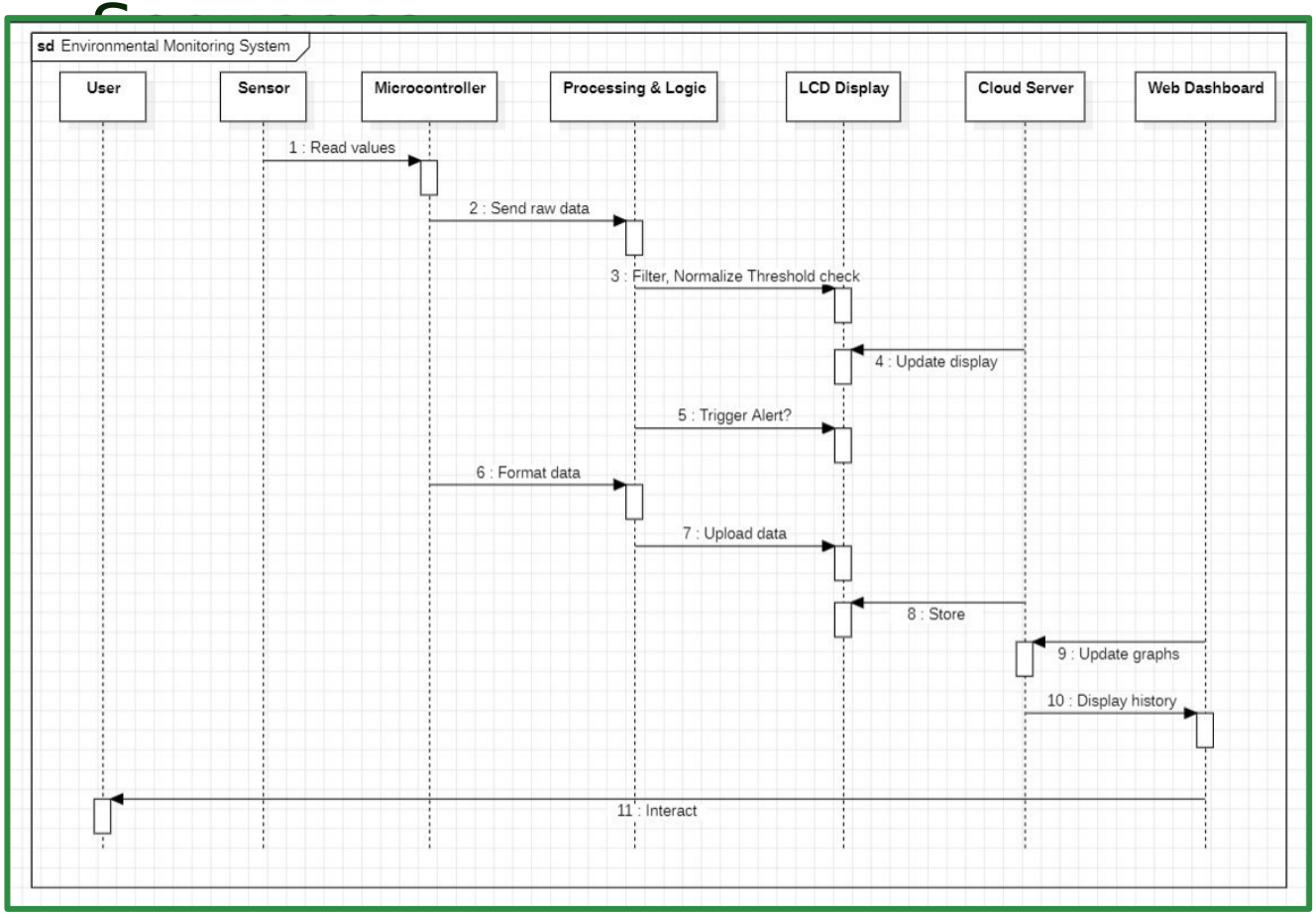
Responsibilities:
-Transmit data to cloud/server or other devices -Handle data formats (JSON)
Technologies:
-Wi-Fi (ESP32)
-LoRa (for long

Responsibilities:
-Filter and normalize sensor data -Perform threshold detection (trigger alert if CO_2 > 1000 ppm)
-Local decision-making logic (turn on fan if temperature > 30°C) **Technologies:**
-Microcontroller -
Python (if using Raspberry Pi)
Rule-based engine

Responsibilities:
-Interface with physical sensors (temperature, humidity, CO_2)
-Collect raw data in real-time
Technologies/Devices:
-DHT11, MQ135, BMP280, etc.
-Microcontroller

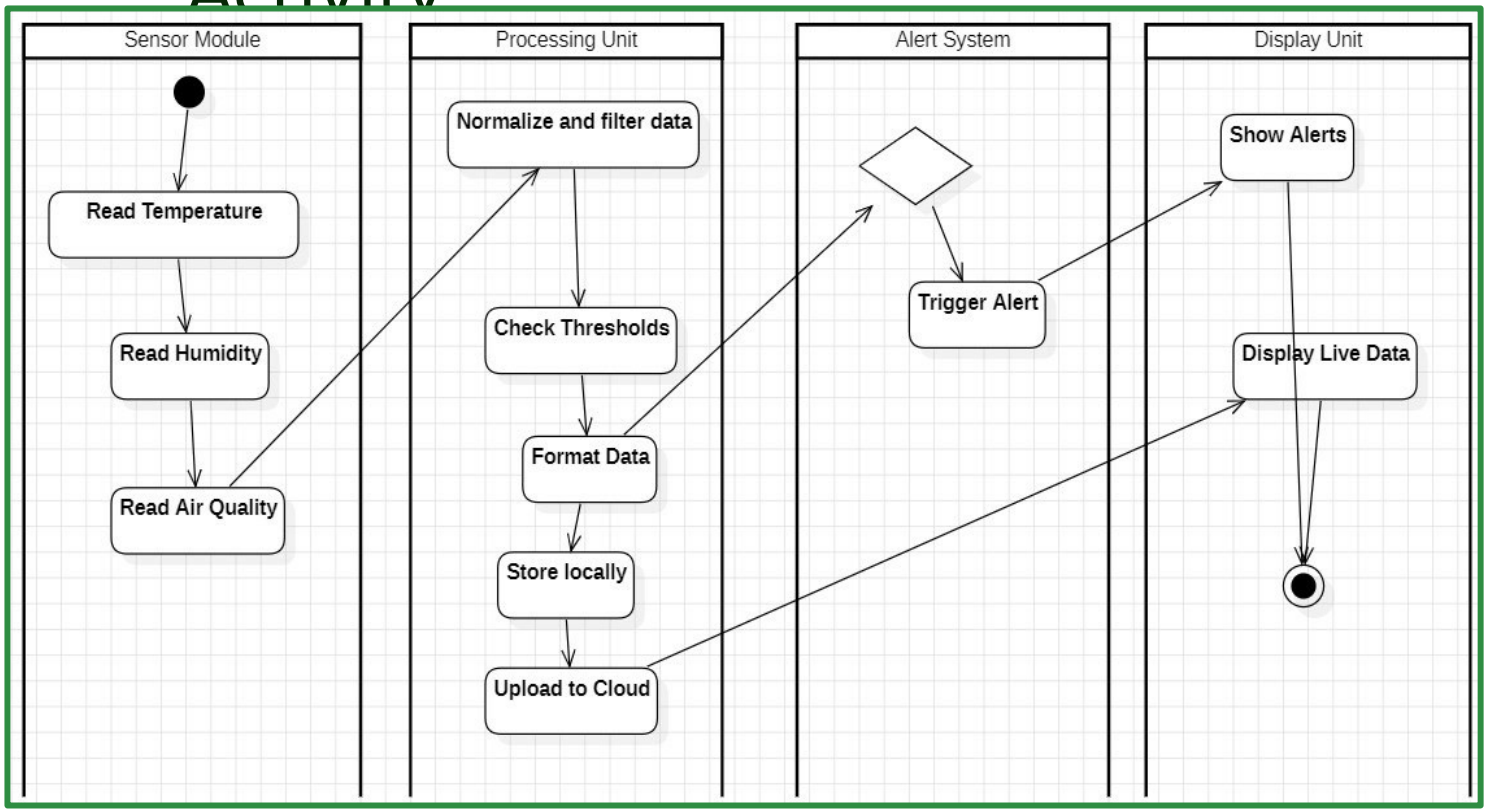
Sensitivity:
Internal

8. Behavioral Modeling –



Sensitivity:
Internal

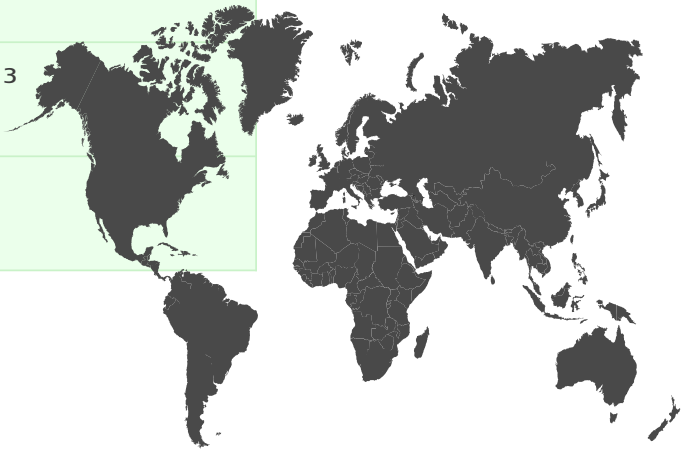
8. Behavioral Modeling – Activity:



Sensitivity: Internal

8. Data design:

Parameter	Type	Range	Units
Temperature	Float	-40 to +85	°C
Humidity	Float	0 to 100	%RH
CO_2 Level	Float	400 to 5000	ppm
PM2.5 Level	Float	0 to 500	µg/m³
Noise Level	Float	30 to 120	dB



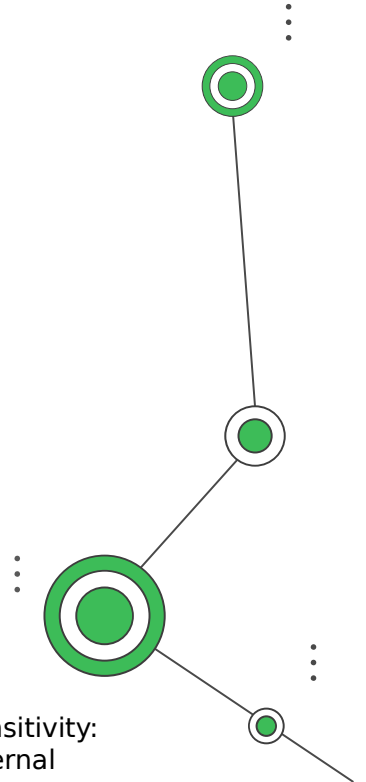
Sensitivity:
Internal

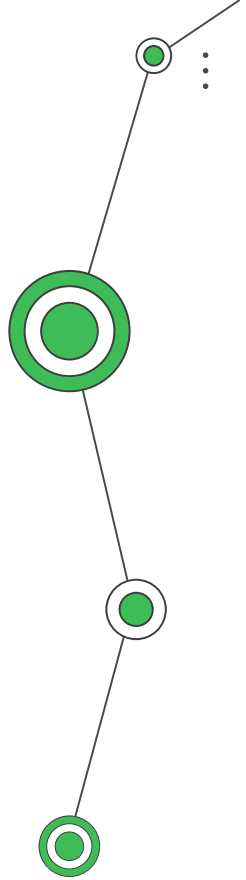
9. User Interface Mockups made with figma:

Sensitivity:
Internal

Thanks!

Do you have any questions?





Sensitivity:
Internal